Refine Search

Search Results -

Terms	Documents
5907847.uref.	12

Database: US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search:

Refine Search

Recall Text

Clear

Interrupt

Search History

DATE: Thursday, September 29, 2005 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L28	5907847.uref.	12	L28
L27	5560005.uref.	163	L27
L26	5809297.uref.	10	L26
L25	5809297.uref.	10	L25
L24	(5418950 | 5295256 | 5307499 | 5212787 | 5278946 | 5263167 | 4930071 | 5195178 | 5499371)![PN]	18	L24
L23	('5809297')[PN]	2	L23
L22	5809297.pn.	2	L22
L21	(5317742 | 5136716 | 5408619 | 5426747 | 5093911)![PN]	10	L21
L20	('5560005')[PN]	2	L20
L19	5560005.pn.	2	L19
L18	5600005.pn.	2	L18
L17	(5566330 | 5295256 | 5644764 | 5577251 | 5291583)![PN]	10	L17
L16	('5907847')[PN]	2	L16
L15	5907847.pn.	2	L15

L14	707/104.1	4797	L14
L13	707.clas.	29589	L13
L12	707/103r	1283	L12
L11	707/102	6366	L11
L10	707/100	6681	L10
L9	707/10	10558	L9
L8	707/4	4319	L8
L7	707/3	7460	L7
L6	707/2	4647	L6
L5	707/1	7062	L5
L4	L3 and (object with table with view or object near table near view or object adj table adj view)	62	L4
L3	L2 and (metadata or meta with data or meta near data)	590	L3
L2	L1 and (relation$ or relationale or relation) near (data with base or database)	2146	L2
L1	(object-oriented or object near oriented) near (data with base or database)	3370	L1

END OF SEARCH HISTORY

Refine Search

Search Results -

Terms	Documents
L12 and 707/103r	19

Database: US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search:

Refine Search

Recall Text | Clear | Interrupt

Search History

DATE: Thursday, September 29, 2005 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
	DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR		
L14	l12 and 707/103r	19	L14
L13	l6 and l12	154	L13
L12	L11 and ("object id" or object with id or object near id or "object identifier" or object near identifier or object with identifier or object adj identifier or object adj identifi$)	154	L12
L11	L10 and view	313	L11
L10	L9 and (metadata or meta with data or meta near data or meta adj data)	380	L10
L9	L8 and tables	961	L9
L8	L7 and (object-oriented or object near oriented or object adj oriented) near (data with base or database)	1054	L8
L7	L6 and (relational or relation) near (data with base or database or data adj base)	6664	L7
L6	707.clas.	29589	L6
L5	345/348	1597	L5
L4	345.clas.	69345	L4
L3	711/203	1416	L3
L2	711.clas.	26993	L2
L1	707/101	4083	L1

Freeform Search

Database: US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Term:

Display: 10 **Documents in Display Format:** - **Starting with Number** 1

Generate: ○ **Hit List** ◉ **Hit Count** ○ **Side by Side** ○ **Image**

Search Clear Interrupt

Search History

DATE: Thursday, September 29, 2005 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L7	"kooi,robert".in.	0	L7
L6	"cheng, tze-pin".in.	1	L6
L5	"wijaya, joyo".in.	8	L5
L4	"hong, chin".in.	53	L4
L3	"haon, chin".in.	0	L3
L2	"nori, anil".in.	24	L2
L1	"oracle corporation".as.	453	L1

END OF SEARCH HISTORY

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 8 of 8 returned.

☐ 1. Document ID: US 20010047285 A1

Using default format because multiple data bases are involved.

L5: Entry 1 of 8 File: PGPB Nov 29, 2001

PGPUB-DOCUMENT-NUMBER: 20010047285
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20010047285 A1

TITLE: Scheduling delivery of products via the internet

PUBLICATION-DATE: November 29, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Borders, Louis H.	Palo Alto	CA	US	
Bhargava, Sunil	Hillsborough	CA	US	
Wijaya, Joyo	Menlo Park	CA	US	
Nijhawan, Sandeep	San Jose	CA	US	

US-CL-CURRENT: 705/8

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

☐ 2. Document ID: US 6266673 B1

L5: Entry 2 of 8 File: USPT Jul 24, 2001

US-PAT-NO: 6266673
DOCUMENT-IDENTIFIER: US 6266673 B1

TITLE: Performing operations on objects in a database system in a response to a request that specifies references that indicate where the objects reside

DATE-ISSUED: July 24, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Hong; Chin-Heng	Hillsborough	CA		
Thakur; Sudheer	Belmont	CA		
Nori; Anil	Fremont	CA		
Wijaya; Joyo	Menlo Park	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Oracle Corporation	Redwood Shores	CA			02

APPL-NO: 09/ 589601 [PALM]
DATE FILED: June 7, 2000

PARENT-CASE:
RELATED APPLICATION This application is a divisional application of U.S. application Ser. No. 08/961,740, entitled "References That Indicate Where Global Database Objects Reside", filed on Oct. 31, 1997 now U.S. Pat. No. 6,134,558 by Chin-Heng Hong, Sudheer Thakur, Anil Nori, and Joyo Wijaya.

INT-CL: [07] G06 F 17/30

US-CL-ISSUED: 707/103; 707/3, 707/10, 707/101, 707/102
US-CL-CURRENT: 707/103R; 707/10, 707/101, 707/102, 707/3

FIELD-OF-SEARCH: 707/10, 707/101, 707/102

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
5261098	November 1993	Katin et al.	707/1
5418888	May 1995	Alden	706/48
5551027	August 1996	Choy et al.	
5560005	September 1996	Hoover et al.	707/10
5581758	December 1996	Burnett et al.	707/103
5594899	January 1997	Knudsen et al.	707/2
5724575	March 1998	Hoover et al.	707/10
5729730	March 1998	Wlaschin et al.	707/3
5740440	April 1998	West	395/704
5764977	June 1998	Oulid-Aissa et al.	707/10
5978791	November 1999	Farber et al.	707/2
6108664	August 2000	Nori et al.	707/103
6134558	October 2000	Hong et al.	707/103

ART-UNIT: 271

PRIMARY-EXAMINER: Alam; Hosain T.

ASSISTANT-EXAMINER: Corrielus; Jean M.

ATTY-AGENT-FIRM: Hickman Palermo Truong & Becker LLP Bingham; Marcel K.

ABSTRACT:

A mechanism is described for processing requests to specify operations to database objects. A request to perform an action on a set of multiple objects is received by a database system. The request includes references to each object in the set, each reference indicating a table where the respective object resides. The reference is used to locate the object, and once located, the action is performed on the object. The reference may indicate a table using a unique table id not used in any of a plurality of databases to identify a table. The action request may be to modify the object, the references may include references to objects that reside in different database systems.

15 Claims, 15 Drawing figures

☐ 3. Document ID: US 6253226 B1

L5: Entry 3 of 8 File: USPT Jun 26, 2001

US-PAT-NO: 6253226
DOCUMENT-IDENTIFIER: US 6253226 B1

TITLE: Duration-based memory management of complex objects

DATE-ISSUED: June 26, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chidambaran; Lakshminarayanan	Fremont	CA		
Krishnaswamy; Srinath	Fremont	CA		
Wijaya; Joyo	Menlo Park	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Oracle Corporation	Redwood Shores	CA			02

APPL-NO: 09/ 103547 [PALM]
DATE FILED: June 24, 1998

PARENT-CASE:
RELATED APPLICATIONS This application is related to the commonly assigned, U.S. patent application Ser. No. 09/103,548, entitled "Memory Management of Complex Objects Returned from Procedure Calls," filed on Jun. 24, 1998 by Lakshminarayanan Chidambaran, the contents of which are hereby incorporated by reference herein.

INT-CL: [07] G06 F 9/00

US-CL-ISSUED: 709/104; 711/5, 711/102, 711/103, 711/1, 707/101, 707/103
US-CL-CURRENT: 718/104; 707/101, 707/103Y, 711/1, 711/102, 711/103, 711/5

FIELD-OF-SEARCH: 709/100, 709/101, 709/102, 709/103, 709/104, 707/338, 707/339, 707/348, 707/101, 707/102, 711/1, 711/5, 711/11, 711/102, 711/103, 711/203

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
5649139	July 1997	Weinreb et al.	711/202
5742793	April 1998	Sturges et al.	711/152
5822590	October 1998	Gupta	717/5
5838977	November 1998	Gupta	717/5
6047280	April 2000	Ashby et al.	707/2

ART-UNIT: 211

PRIMARY-EXAMINER: Banankhah; Majid A.

ATTY-AGENT-FIRM: Ditthavong & Carlson, P.C.

ABSTRACT:

Memory for complex objects is maintained in pools of dynamic memory on a "per-duration" basis. Each duration is assigned its own area or areas of the heap, and all the memory allocation for a specific duration comes from those assigned areas of the heap. Memory allocation for a complex object is performed with respect to a single duration and, hence, memory is allotted for the complex object from the corresponding memory pool. When a duration is terminated, the memory allocated for its corresponding heap is freed, thereby releasing memory for all the complex object using the memory from the memory pool for that duration. Management of other resources for complex objects such as opening and closing files may also be duration-based. In one aspect, the memory management of complex objects is located in an automatically generated client stub routine for a remote procedure call. Accordingly, the interface description language (IDL) for the remote procedure call is extended to incorporate the duration idea for out parameters.

36 Claims, 7 Drawing figures

☐ 4. Document ID: US 6134558 A

L5: Entry 4 of 8 File: USPT Oct 17, 2000

US-PAT-NO: 6134558
DOCUMENT-IDENTIFIER: US 6134558 A

TITLE: References that indicate where global database objects reside

DATE-ISSUED: October 17, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Hong; Chin-Heng	Hillsborough	CA		
Thakur; Sudheer	Belmont	CA		
Nori; Anil	Fremont	CA		
Wijaya; Joyo	Menlo Park	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Oracle Corporation	Redwood Shores	CA			02

APPL-NO: 08/ 961740 [PALM]
DATE FILED: October 31, 1997

INT-CL: [07] G06 F 17/30

US-CL-ISSUED: 707/103; 707/10, 707/101, 707/102
US-CL-CURRENT: 707/103R; 707/10, 707/101, 707/102

FIELD-OF-SEARCH: 707/103, 707/101, 707/102, 707/3, 707/10

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
5261098	November 1993	Katin et al.	707/1
5551027	August 1996	Choy et al.	707/201
5560005	September 1996	Hoover et al.	707/10
5581758	December 1996	Burnett et al.	707/103
5594899	January 1997	Kundsen et al.	707/2
5724575	March 1998	Hoover et al.	707/10
5729730	March 1998	Wlaschin et al.	707/3
5978791	November 1999	Farber et al.	707/2

ART-UNIT: 277

PRIMARY-EXAMINER: Homere; Jean R.

ATTY-AGENT-FIRM: Hickman Palermo Truong & Becker, LLP Hickman; Brian D. Bingham; Marcel K.

ABSTRACT:

A method and apparatus for generating references to a set of objects which reside in a plurality databases is described. Each object is associated with a table from a plurality of tables that are contained in the plurality of databases. An object id is associated with each object; the object id uniquely identifies the object relative to the objects in the set of objects. A table id is associated with each table; the table id uniquely identifies the table relative to tables in the plurality of tables. A table containing an object is located based on the table id associated with the table, and the object is located in the table based on the object id associated with the object. A table mapping is generated. The table mapping maps a set of tables to databases associated with the set of tables. The set of tables are from the plurality of tables. References to objects from the set of object are generated. Each reference comprises data that identifies an object. The reference contains data representing the object id of the referenced object, the object referred to by the reference. The reference also contains data representing the table id of the table containing the referenced object. An object is located based on the table mapping and the reference referring to the object. The table containing the object is located based on the data in the reference, the data representing the table id associated with the table containing the object.

43 Claims, 15 Drawing figures

5. Document ID: US 6108664 A

L5: Entry 5 of 8 File: USPT Aug 22, 2000

US-PAT-NO: 6108664
DOCUMENT-IDENTIFIER: US 6108664 A

TITLE: Object views for relational data

DATE-ISSUED: August 22, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Nori; Anil	San Jose	CA		

Hong; Chin	Hillsborough	CA
Wijaya; Joyo	Menlo Park	CA
Cheng; Tze-Pin	Milpitas	CA
Kooi, deceased; Robert P.	late of Lafayette	CA

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Oracle Corporation	Redwood Shores	CA			02

APPL-NO: 08/ 962415 [PALM]
DATE FILED: October 31, 1997

PARENT-CASE:
RELATED APPLICATION The present application is related to U.S. patent application Ser. No. 08/961,740, entitled "REFERENCES TO GLOBAL DATABASE OBJECTS", pending filed by Chin-Heng Hong, Sudheer Thakur, Anil Nori, Joyo Wijaya, on the equal day herewith, the contents of which are incorporated herein by reference. U.S. patent application Ser. No. 08/961,794, entitled "APPARATUS AND METHOD FOR PICKLING DATA", pending filed by John Wiesz, on the equal day herewith, the contents of which are incorporated herein by reference. U.S. patent application Ser. No. 08/962,409, entitled "APPARATUS AND METHOD FOR OBJECT REPRESENTATION AND STORAGE IN A RELATIONAL DATABASE SYSTEM", pending filed by Anil Nori, John Weisz, Vikas Arora, Subramanian Muralidhar, on the equal day herewith, herein referred to as, the contents of which are incorporated herein by reference. U.S. patent application Ser. No. 08/962,535, entitled "APPARATUS AND METHOD FOR STORAGE OF OBJECT COLLECTIONS IN A DATABASE SYSTEM", pending filed by Anil Nori, Vishu Krishnamurthy, Vikas Arora, Srinath Krishnaswamy, on the equal day herewith, the contents of which are incorporated herein by reference. U.S. patent application Ser. No. 08/962,416, entitled "APPARATUS AND METHOD FOR NULL REPRESENTATION IN DATABASE OBJECT STORAGE", pending filed by Anil Nori, John Weisz, Subramanian Muralidhar, on the equal day herewith, the contents of which are incorporated herein by reference.

INT-CL: [07] G06 F 17/30

US-CL-ISSUED: 707/103; 707/102, 707/104
US-CL-CURRENT: 707/103R; 707/102, 707/104.1

FIELD-OF-SEARCH: 707/103, 707/2, 707/10, 707/101, 707/102, 707/104, 711/203, 345/348

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
5335346	August 1994	Fabbio	395/600
5426747	June 1995	Weinreb et al.	711/203
5448727	September 1995	Annevelink	707/101
5560005	September 1996	Hoover et al.	707/10
5615367	March 1997	Bennett et al.	707/102
5680563	October 1997	Edelman	345/348
5724575	March 1998	Hoover et al.	707/10
5752018	May 1998	Sheffield	707/2
5765159	July 1998	Srinivasan	707/102
5809297	September 1998	Kroenke et al.	707/102
5815710	September 1998	Martin et al.	709/303
5832498	November 1998	Exertier	707/103
5907846	May 1999	Brener et al.	707/103

5926819	July 1999	Doo et al.	707/104
5937409	August 1999	Wetherbee	707/103
5995973	November 1999	Daudenarde	707/103

ART-UNIT: 271

PRIMARY-EXAMINER: Alam; Hosain T.

ASSISTANT-EXAMINER: Colbert; Ella

ATTY-AGENT-FIRM: Hickman Palermo Truong & Becker, LLP Hickman; Brian D. Bingham; Marcel K.

ABSTRACT:

A method and apparatus for presenting and modifying data from a set of tables in a database is provided. A view that is defined is based on a set of one or more tables that may include relational tables or object tables. The view defines a presentation of data from the one or more tables as a set of objects that reside in the database. Data is read from the one or more rows of the tables based on the view, and is presented as a set of objects that reside in the database. An object id that is based on data from the one or more rows is generated and associated with each object presented. The view may specify which columns from the one or more tables contain values used to generate the object ids. A trigger may associated with the view. The set of objects presented may be presented as objects having an attribute that is a column object. Column objects include user specified object types, collection objects (e.g. nested tables and variable arrays), or references to objects.

39 Claims, 20 Drawing figures.

☐ 6. Document ID: US 6006234 A

L5: Entry 6 of 8 File: USPT Dec 21, 1999

US-PAT-NO: 6006234
DOCUMENT-IDENTIFIER: US 6006234 A

TITLE: Logical groupings within a database

DATE-ISSUED: December 21, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Govindarajan; Rajagopalan	Fremont	CA		
Kotsovolos; Susan	Belmont	CA		
Krishnan; Ramkumar	Nashua	NH		
Wijaya; Joyo	Menlo Park	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Oracle Corporation	Redwood Shores	CA			02

APPL-NO: 08/ 962531 [PALM]
DATE FILED: October 31, 1997

INT-CL: [06] G06 F 17/30

US-CL-ISSUED: 707/103; 707/3, 707/4, 707/10, 707/101, 707/104
US-CL-CURRENT: 707/103R; 707/10, 707/101, 707/104.1, 707/3, 707/4

FIELD-OF-SEARCH: 707/3, 707/4, 707/10, 707/101, 707/102, 707/103, 707/104, 370/249, 709/303, 341/65

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
5608720	March 1997	Biegel	370/249
5754841	May 1998	Carino	707/3
5799306	August 1998	Sun	707/10
5799309	August 1998	Srinivasan	707/102

ART-UNIT: 271

PRIMARY-EXAMINER: Black; Thomas G.

ASSISTANT-EXAMINER: Mizrahi; Diane D.

ATTY-AGENT-FIRM: McDermott, Will & Emery

ABSTRACT:

A method, system and computer-readable medium is provided for grouping database objects into logical groupings in order to simplify administrative and other operations that need to be performed by the database server. Such operations can be performed once at the logical group level for a group of related objects, as opposed to at the individual database object level. For increased flexibility, the logical groupings need not dictate the format, schema or location of their members. A hierarchy may be established between the logical groupings, where child groupings inherit some or all of the properties of the parent groupings. A correspondence may be established between some groupings and operating system directories, allowing identifiers associated with the groupings to be used as aliases for the full operating system paths to the corresponding directories.

25 Claims, 2 Drawing figures

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Draw Desc | Image

---

☐ 7. Document ID: US 5887275 A

L5: Entry 7 of 8 File: USPT Mar 23, 1999

US-PAT-NO: 5887275
DOCUMENT-IDENTIFIER: US 5887275 A

TITLE: Method and apparatus for locating a first set of data using a global identifier and a virtual addressing data structure

DATE-ISSUED: March 23, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Nguyen; Tin Anh	Danville	CA		
Wijaya; Joyo	Menlo Park	CA		
Boonleungtomnu; John	San Jose	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Oracle Corporation	Redwood Shores	CA			02

APPL-NO: 08/ 598517 [PALM]
DATE FILED: February 8, 1996

INT-CL: [06] G06 F 12/08

US-CL-ISSUED: 711/206; 707/100
US-CL-CURRENT: 711/206; 707/100

FIELD-OF-SEARCH: 711/203, 711/206, 707/103, 707/100, 395/683

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4660142	April 1987	Clancy et al.	711/202
5522077	May 1996	Cuthbert et al.	395/683
5581765	December 1996	Munroe et al.	395/682
5615362	March 1997	Jensen et al.	707/103
5615363	March 1997	Jenness	707/103
5634124	May 1997	Khoyi et al.	707/103

OTHER PUBLICATIONS

Andrea H. Skarra et al., "The Management of Change Types in an Object-Oriented Database," OOPSLA '86 Proceedings, Brown University, Department of Computer Science, Providence RI, pp. 483-495 (Apr. 1986).
John F. Roddick, "Schema Evolution in Database Systems--An Annotated Bibliography," SIGMOD Record, vol. 21, No. 4, Dec. 1992, School of Computer and Information Science, University of South Australia, The Levels, SA 5095, South Australia, pp. 35-39.
Elisa Bertinno et al., "Object-Oriented Database Management Systems: Concept and Issues," Computer, vol. 24, No. 4, 15 pages (Apr. 1991).
J. Eliot, B. Moss, "Working with Persistant Objects: To Swizzle or Not to Swizzle," IEEE Transactions on Software Engineering, vol. 18, No. 8, Aug. 1992.
Charles W. Krueger, "Software Reuse," ACM Computing Surveys, vol. 24, No. 2, 53 pages (Jun. 1992).
K. Narayanaswamy et al., "An Incremental Mechanism for Schema Evolution in Engineering Domains," 4th Int'l Conference on Data Engineering, Los Angeles, California pp. 294-301 (1988).

Simon Monk et al., "Schema Evolution in OODBs Using Class Versioning," SIGMOD Record, vol. 22, No. 3, pp. 16-22 (Sep. 1993).
J.F. Roddick, "Dynamically Changing Schemas Within Database Models," Australian Computer Journal, vol. 23, No. 3, pp. 105-109 (1991).
Jay Banerjee et al., "Semantics and Implementation of Schema Evolution in Object-Oriented Databases," ACM 0-89791-236-5, pp. 311-322 (1987).
D. Beech et al., "Generalized Version Control in an Objected-Oriented Database," 4th IEEE-Int'l Conference on Data Engineering, Los Angeles, California, pp. 14-22 (1988).
Simon Gibbs et al., "Class Management for Software Communities," Communications of the ACM,

vol. 33, No. 9, pp. 90-103 (1990).
G.T. Nguyen et al., "Schema Evolution in Objected-Oriented Database Systems," Data & Knowledge Engineering 4, pp. 43-67 (1989).
B.S. Lerner et al., "Beyond Schema Evolution to Database Reorganization," SIGPLAN Notices, vol. 25, No. 10, pp. 67-76 (1990).
D.Jason Penney et al., "Class Modification in the GemStone Object-Oriented DBMS," SIGPLAN Notices (Proceedings of OOPSLA 1987), vol. 22, No. 12, pp. 111-117 (1987).
L. Tan et al., "Meta Operations for Type Management in Object-Oriented Databases--a Lazy Mechanism for Schema Evolution," 1st Int'l Conf., Kyoto, Japan, pp. 241-258 (1989).
Stanley B. Zdonik, "Object-Oriented Type Evolution," Advances in Database Programming Languages, ACM Press, New York, pp. 277-288 (1990).

ART-UNIT: 271

PRIMARY-EXAMINER: Chan; Eddie P.

ASSISTANT-EXAMINER: Portka; Gary J.

ATTY-AGENT-FIRM: McDermott, Will & Emery

ABSTRACT:

A method and apparatus are provided for locating a object based on a reference to the object. An application determines whether the reference has previously been used to locate the object. If the reference has previously been used to locate the object, then a data structure referred to as a "tombstone" that has been associated with the object is located based on a first virtual memory address that is stored in the reference. Once the tombstone has been located, a first pseudo-timestamp that is stored in the reference is compared to a second pseudo-timestamp that is stored in the tombstone. If the first pseudo-timestamp matches the second pseudo-timestamp, then the object is located based on a second virtual memory address that is stored in the tombstone. If the first pseudo-timestamp does not match the second pseudo-timestamp or if the reference has not been previously used to locate the object, then the object is located based on the identifier stored in the reference. During the process of locating the object based on the object identifier, the virtual address of the tombstone associated with the object is stored in the reference, and the pseudo-timestamp stored in the tombstone is stored in the reference. When tombstones are reused, the pseudo-timestamp within the tombstone is incremented.

23 Claims, 7 Drawing figures

---

8. Document ID: US 5787300 A

L5: Entry 8 of 8 File: USPT Jul 28, 1998

US-PAT-NO: 5787300
DOCUMENT-IDENTIFIER: US 5787300 A
**** See image for Certificate of Correction ****

TITLE: Method and apparatus for interprocess communications in a database environment

DATE-ISSUED: July 28, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Wijaya; Joyo	Menlo Park	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Oracle Corporation	Redwood City	CA			02

APPL-NO: 08/ 584910 [PALM]
DATE FILED: January 11, 1996

PARENT-CASE:
This application is a continuation of Ser. No. 08/150,592 filed Nov. 10, 1993, now abandoned.

INT-CL: [06] G06 F 13/00

US-CL-ISSUED: 395/800.01; 395/800.26, 370/282, 370/299, 364/DIG.1
US-CL-CURRENT: 712/1; 370/282, 370/299, 712/26

FIELD-OF-SEARCH: 395/800.01, 395/800.26, 370/282, 370/299, 364/DIG.1

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4862350	August 1989	Orr et al.	395/250
5148527	September 1992	Basso et al.	395/325
5214759	May 1993	Yamoaka et al.	395/325
5247626	September 1993	Firoozmand	395/250
5265250	November 1993	Andrade et al.	395/650
5287453	February 1994	Roberts	395/200
5287456	February 1994	Rhode et al.	395/200
5313638	May 1994	Ogle et al.	395/725
5321808	June 1994	Rupp	395/164
5325492	June 1994	Bonevento	395/325
5329619	July 1994	Page et al.	395/200.01
5371850	December 1994	Belsan et al.	395/200

ART-UNIT: 232

PRIMARY-EXAMINER: Bowler; Alyssa H.

ASSISTANT-EXAMINER: Nguyen; Dzung C.

ATTY-AGENT-FIRM: Fliesler Dubb Meyer & Lovejoy LLP

ABSTRACT:

The present invention provides interprocess communication in a DBMS. The present invention provides the ability for these processes to communicate with other DBMS processes or processes external to the DBMS. A pipe is implemented as an object of the general purpose object cache. The general purpose object cache resides in the systems shared memory space. It is concurrently accessible by many sessions, or processes. A pipe is located in a shared global memory area. The present invention provides the ability to send a message (i.e., record) to a pipe, and receive a message (i.e., record) from a pipe. A pipe is located in shared memory. Shared memory can contain multiple pipes. Each pipe is comprised of a linked list of records, and linked list of sessions, an exclusivity indicator, and a session waiting indicator. Multiple sessions can access the same pipe, and each pipe can contain multiple messages. A message is sent by a sending session to a local buffer. The contents of the local buffer is sent to a pipe. The contents of a pipe, one or more records, can be accessed by getting a record from the pipe and

placing it in a local buffer. The contents of the local buffer can be accessed by the requesting session.

26 Claims, 25 Drawing figures

Terms	Documents
"wijaya, joyo".in.	8

L4: Entry 57 of 62 File: USPT Oct 19, 1999

US-PAT-NO: 5968115
DOCUMENT-IDENTIFIER: US 5968115 A

TITLE: Complementary concurrent cooperative multi-processing multi-tasking processing system (C3M2)

DATE-ISSUED: October 19, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Trout; Ray C.	Houston	TX		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Complementary Systems, Inc.	Houston	TX			02

APPL-NO: 08/ 794045 [PALM]
DATE FILED: February 3, 1997

INT-CL: [06] G06 F 9/46

US-CL-ISSUED: 709/107; 395/840
US-CL-CURRENT: 718/107; 710/20

FIELD-OF-SEARCH: 395/670-675, 395/840, 395/841, 395/827, 709/107

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5016166	May 1991	Van Loo et al.	364/200
☐	5113500	May 1992	Talbott et al.	395/325
☐	5257372	October 1993	Furtney et al.	395/650
☐	5307485	April 1994	Bordonaro et al.	395/600
☐	5317693	May 1994	Cuenod et al.	395/275
☐	5402350	March 1995	Kline	364/368
☐	5450581	September 1995	Bergen et al.	355/600
☐	5566349	October 1996	Trout	395/840

OTHER PUBLICATIONS

Heinrich et al, The Performance Impact of Flexibility in the Stanford Flash Multiprocessor, ACM Mar. 1994.
Heinlein et al, Integration of Message Passing and Shared Memory in the Stanford Flash Multiprocessor ACM Mar. 1994.
Arthur, Lowell Jay, Improving Software Quality, 1993, Table of Contents.
Arthur, Lowell Jay, Improving Software Quality, Chap 13: Rapid Evolutionary Devel.
IBM Dictionary of Computing, Aug. 1993 Preface.
Booch, Grady, Object Oriented Design With Applications, 1991, Table Content.
Connell et al, Structured Rapid Proto Typing, 1989, Table of Contents.
Yourdan, Edward, Decline and Fall of the American Programmer, 1992 Table Content.

ART-UNIT: 275

PRIMARY-EXAMINER: Toplu; Lucien U.

ATTY-AGENT-FIRM: Casperson; John R.

ABSTRACT:

The system concept of the C3M2 System is to have the capability of providing a Process for each major processing step of automated data processing, i.e. if you have four steps then you need a minimum of four but it could be 8 or 12 or 16 processes. The four major complementary functions encompass the four major functions of data processing (Input/Output, Data Computation, Storage and User I/F). The system shall be Multi-tasking for each step. Source headers, link lists and entity or object identifiers are the methods that shall be used for identity of the different classes, types and objects for the variety of data in the system. The source and data type are contained in the source header. The class and type identity are contained in the object identifiers. The multi-tasking would be by schedule (interleaved by priority). This was selected instead of cycle sharing for improved concurrency.

19 Claims, 22 Drawing figures

Previous Doc Next Doc Go to Doc#

L4: Entry 57 of 62 File: USPT Oct 19, 1999

DOCUMENT-IDENTIFIER: US 5968115 A
TITLE: Complementary concurrent cooperative multi-processing multi-tasking processing system (C3M2)

Detailed Description Text (18):
Memory Management: Memory allocations made during the planning phase for all memory users. Memory segments are divided into 1/1024 segments of dedicated total active memory. The data archive memory occupies the majority of the memory and is in a Relational Database Format. The other memory segments provide the required memory storage elements of the Complementary processing's, systems Software and utilities. Each Processes writes to its memory, Load/Locked. Other processes Read or Exchange. The memory also serves as an interface between the Processes. The memory is identified as the Congruent Memory (CM).

Detailed Description Text (22):
Object: In Structured query language, anything that can be created, accessed or manipulated with Structured query language statements, such as databases, tables, records, views or indexes.

Detailed Description Text (42):
One of the features of a preferred embodiment of the invention is the storage of data in a relational database for fast retrieval and response. This functionality is provided by the DS process 40. A DS CSM is preferably operably associated with the DS process. An archive memory 5.4 is provided for storing data for archive from the DS process. The DS instructions and means for causing the DS process to retrieve data from IO memory 5.1, 5.2, 5.6 causes the DS process to retrieve such data into the DS CSM. The DS instructions further include a routine to retrieve dynamic data from the DS CSM, a routine to retrieve static data from the DS CSM, a routine to retrieve calculated data from the DS CSM, and a routine to retrieve the identifiers for the dynamic data, the static data, and the calculated data from the DS CSM. The identifiers are converted to record numbers by a routine in the DS instructions. The DS instructions further include a routine to convert the dynamic data, the static data, the calculated data and the record numbers to a relational data base format for storage as a relational data base in the DS memory and a routine for storing the relational data base in the DS memory. To provide long term storage and accessibility of information in the long term storage, the DS instructions further include a routine for retrieving the relational data base from the DS memory, a routine for storing the thus retrieved relational data base in the archive memory, a routine for transferring the relational data base from the archive memory to an archive media for storage, and a routine for transferring a relational data base from an archive media to the archive memory.

Detailed Description Text (50):
To prepare for setting up the relational database, the IO process identifies the dynamic data and the static data and transfers the dynamic data as IO output to a first portion of the IO memory for storage and the static data as IO output to a second portion of the IO memory for storage. The multicharacteristic identifiers are transferred as IO output to a third portion of the IO memory for storage. The DS process completes setting up the relational database. An IO memory output is retrieved from the first portion of the IO memory, the second portion of the IO memory, and the third portion of the IO memory. The retrieved IO memory outputs are received by the DS process. The received IO memory outputs are reformatted in the DS process which produces a reformatted DS output. The reformatted DS output is transferred to the DS memory for storage.

Detailed Description Text (54):
Data records for use by the system are Relational Database (RDB) formats for the entities,

objects and attributes in a Flat File. The Flat File is defined as an array of data records. The files are by classes of system elements. The RDB files to reside in the Congruent Memory (CM) for Real Time access and are stored in the Archive Media (AM) on a cyclic basis that is determined in the planning phase. The latency time for access of the AM will be in the millisecond range. The cyclic time for exchange of data from the CM to the AM would be in the Hours-Type time period. Data files for making up the Object data records are initial or identity Data, Calculated Data for object performance & etc., Reference Data from the External Data base managers and Knowledgebase, Link List and Data Source data for each record. The entire record then becomes a part of the individual records for each object file number for the Real Time Archive data set.

Detailed Description Text (184):
Access to metadata--The C3M2 system shall maintain the integrity of its database by prohibiting operations that would corrupt the system, e.g. certain updates to metadata.

Detailed Description Text (236):
Persistent objects--The C3M2 system shall provide database management support in accordance with the concept of Object Oriented Database Management (OODM).

Previous Doc Next Doc Go to Doc#

L4: Entry 59 of 62 File: USPT Feb 16, 1999

US-PAT-NO: 5872973
DOCUMENT-IDENTIFIER: US 5872973 A

TITLE: Method for managing dynamic relations between objects in dynamic object-oriented languages

DATE-ISSUED: February 16, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Mitchell; David C.	South Orem	UT		
Anderson; Kelly L.	Provo	UT		
Osman; Andrew V.	Provo	UT		
Mitchell; Dale K.	Provo	UT		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Viewsoft, Inc.	Provo	UT			02

APPL-NO: 08/ 548536 [PALM]
DATE FILED: October 26, 1995

INT-CL: [06] G06 F 9/44

US-CL-ISSUED: 395/685; 395/683, 395/702
US-CL-CURRENT: 719/332; 717/108, 717/116, 719/317, 719/330

FIELD-OF-SEARCH: 395/685, 395/710, 395/614, 395/683, 395/701, 395/702, 395/708, 364/284.4, 707/103

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5327562	July 1994	Adcock	395/700
☐	5339430	August 1994	Lundin et al.	395/700
☐	5359721	October 1994	Kempf et al.	395/425
☐	5367633	November 1994	Matheny et al.	395/164
☐	5369766	November 1994	Nakano et al.	395/700
☐	5371891	December 1994	Gray et al.	395/700
☐	5418964	May 1995	Lonner et al.	395/700

☐	5423841	June 1995	Bunke et al.	395/700
☐	5437025	July 1995	Bale et al.	395/600
☐	5485671	January 1996	Stutz et al.	395/700
☐	5515536	May 1996	Corbett et al.	395/700
☐	5619638	April 1997	Duggan et al.	395/703

OTHER PUBLICATIONS

Huy Hguyen et al., OODDM: An Object-Oriented Database Design Model, Apr., 1990, see page 339, right-hand column, line 35; and page 341, left-hand column, line 2.
C. Popien et al., A Concept For An ODP Service Management, IEEE, 1994, see pages 888-897.
Cheng et al, "On The Performance Issues of Object-Based Buffering", Proceedings of the First International Conference on Parallel and Distributed Information Systems, p. 30-37, 4 Dec. 1991.

ART-UNIT: 274

PRIMARY-EXAMINER: Trammell; James P.

ASSISTANT-EXAMINER: Chaki; Kakali

ATTY-AGENT-FIRM: Wilson Sonsini Goodrich & Rosati

ABSTRACT:

A method and system for creating named relations between classes in a dynamic object-oriented programming environment via mappers is disclosed. The mapping objects dynamically bind to the class interfaces of the classes being related. These connections between classes are defined within a visual environment. The relationships can be programmatically attached by name to object instances during program execution. Because these relationships are stored in a resource and are dynamically bound by name to the objects, they can be created and modified without requiring the source code of the objects being associated to be changed. This eliminates hard coded dependencies between objects that impede reuse of the objects in other contexts. The invention requires and takes full advantage of, meta-data, full dynamic binding and probing support in the objects being connected with the invention.

33 Claims, 4 Drawing figures

Previous Doc Next Doc Go to Doc#

L4: Entry 59 of 62 File: USPT Feb 16, 1999

DOCUMENT-IDENTIFIER: US 5872973 A
TITLE: Method for managing dynamic relations between objects in dynamic object-oriented languages

Abstract Text (1):
A method and system for creating named relations between classes in a dynamic object-oriented programming environment via mappers is disclosed. The mapping objects dynamically bind to the class interfaces of the classes being related. These connections between classes are defined within a visual environment. The relationships can be programmatically attached by name to object instances during program execution. Because these relationships are stored in a resource and are dynamically bound by name to the objects, they can be created and modified without requiring the source code of the objects being associated to be changed. This eliminates hard coded dependencies between objects that impede reuse of the objects in other contexts. The invention requires and takes full advantage of, meta-data, full dynamic binding and probing support in the objects being connected with the invention.

Brief Summary Text (29):
meta-data (also referred to as introspection, reflection or run-time type information (RTTI)),

Brief Summary Text (34):
Meta-Data is a database available at runtime (during program execution) that describes the classes used to build the program. It enables objects to be "self-describing" at runtime. When meta-data is extensible this creates an opportunity to have class wide read-only data that may be application specific. This application specific meta-data is called properties, and is an important language feature that is fully utilized in the invention. However, in language systems lacking extensible meta-data, property information can be easily stored in other data structures.

Brief Summary Text (38):
Together, these language features provide objects with the ability to have dynamically reconfigurable input and output during program execution without requiring special behavior on the part of the objects themselves. The term "dynamic language" is defined to mean a language that supports, or has been enhanced to support meta-data, full dynamic binding and probes.

Brief Summary Text (58):
It is also an objective of the present invention to provide a visual programming environment supporting the specification of dynamic linkages (connections) of objects in languages supporting (or enhanced to support) meta-data, full dynamic binding, probing and generic factory method capabilities.

Brief Summary Text (69):
Becoming a client to any class for which the real object is not also a client reduces the semantic purity of a class. A dog may have a relationship to an owner, but a dog should not have a relationship to a database within itself Any relationship to a database should be managed externally to the dog object itself

Brief Summary Text (73):
The main function of the invention is to specify and then instantiate semantic links between objects that have been defined in a dynamic object-oriented language. In the present invention, a dynamic object-oriented language refers to an object-oriented language that supports, or has been enhanced to support, meta-data, dynamic binding and probes. That is, to dynamically link one or more of the members (attributes, functions and properties) of one class to one or more of the members of another class in a language supporting extensible meta-data, full dynamic

binding and probes. This includes dynamically binding to and/or probing members of members of attributes recursively.

Brief Summary Text (86):
A single semantic link involves three objects: The two patron objects being linked, and a third object, the mapper, that does the work involved in setting up, maintaining and ending the semantic link. The mapper is never referenced or visible to the program code of the patron objects. It is built dynamically from a specification stored in the meta-data resource file. Since these semantic links are stored entirely in a resource, and are instantiated dynamically at runtime, the links can be changed without changing the code used to implement the patron classes. There are also helper objects that the mapper calls upon to handle general issues.

Brief Summary Text (91):
In the preferred embodiment, each connection is created for and belongs to a specific class and can only be used with an instance of that class (or a related class). In a sense, the connection is a meta-"member" of a class. Nevertheless, there may also be some advantages to storing some connections separately from class' meta-data.

Brief Summary Text (105):
The invention consists of a set of cooperative systems. The programmer uses a visual interactive builder program to specify connections between objects whose meta-data is available to the builder. In the preferred embodiment, the classes being linked are also created using the same builder that is used to specify connections between the classes. When the programmer finishes building and connecting classes, he saves the meta-data for the classes, including specifications of connections, to a resource file. During program execution, the resource file is read into memory by "edit" and the links are dynamically established by taking advantage of dynamic binding and probing to attach the live objects together according to the blueprint in the resource file.

Brief Summary Text (110):
In the preferred embodiment, the resource file produced by the builder contains a database of meta-data with an entry for each of the programmer defined classes specified in the builder. Connections are stored with one of the classes that is involved in the connection although they could be stored separately in a different implementation. The resource file is a hierarchical attributed data structure known as a table. Tables are capable of storing generic information (similar to a database) in an easily extensible manner. Part of the information stored about a class is a list of all the connections that have been created in the builder to connect that class within itself and to other classes. The information consists of the name of the mapping (each connection for a particular class must have a unique name), the class that this mapping links to and a list of individual links (with properties) that make up the connection. The meta-data for each semantic link stores the attribute, property and function names being connected in the two classes as well as any property information that the mapper requires.

Brief Summary Text (114):
When the libraries are linked with the programmer's code, an executable image results. The program executable consists of the programmer's classes and the resource file (which may be bound to the executable in a resource fork). When the program runs, it loads the resource information file into memory so that the meta-data can be accessed. The connections are stored in the meta-data of the classes being connected. The programmer's classes access the edit function from the runtime library causing the meta-data to be interpreted and the connections are dynamically established during execution of this executable image.

Detailed Description Text (34):
The path object also solves some really difficult situations related to the dynamic nature of objects in a running program. If, for example, a semantic link were made to a field inside of a pointer field, and the pointer field is reassigned to point to a new object, then the mapper has to disconnect its probes to the first object, and reconnect to the new object. The attachments to subfields of subfields is kept track of by keeping a path of field names, for example, fieldA.fieldB.fieldC would mean that fieldC is a member of fieldB, which is a member of fieldA. The path object plants probes on each of these intermediate levels and when an intermediate level changes, it calls the mapper's functions to reset their probes. Mappers must be able to detach from the old object and reattach themselves to the new objects as they change at runtime, but the path object does all the complex bookkeeping. The raw probe machinery is only aware of individual instances to probe, and does not have the contextual ability to detach

and reattach as necessary like the mapper's and paths do. If probes are planted by hand, this functionality may not be obtained automatically. Dynamic binding and meta-data are used to find the new item to plant probes on.

Detailed Description Text (40):
One of the design goals of the system is to make building a new mapper class as easy as possible. Requiring simple mappers deal directly with issues of meta-data and dynamic binding overly complicates their implementation. Type elements are introduced to make access to the patron objects (from the point of view of the mapper) as generic as possible. Each mapper class has two type elements; one for the left side and one for the right side. Of course, the names left and right have no particular meaning to the computer, and are just for human consumption (although there is a correlation to the user interface described later). One major function of the type elements is to have a standard interface that makes functions, properties and lists of members look as much like fields as possible. Having all of these language elements look as similar as possible to the mapper means that mappers do not have to have special purpose code to handle common general cases.

Detailed Description Text (43):
EosTypeElement (301) is the base class for all other type elements. It provides a common base level API for the other type element subclasses. It strives to provide similar services for all the other types, but it most closely resembles a field. Specifically, it tries to make functions and properties act like fields in as many cases as it makes sense to do so. For example, all type elements except for the list element have a function to set data and a function to get data. EosTypeElements (as well as the mappers) should be programmed in a language with the same language requirements (probes, meta-data, dynamic binding and generic factory methods) as the patron objects so they can be created by name, etc. In fact the type elements and the mappers must be in the same executable program, so it is most convenient if they are in the same language.

Detailed Description Text (52):
EosFunctionElement is used when the language element being mapped to is a member function of the patron object. This element has additional abilities to access the function parameter list meta-data information. Functions following these forms:

Detailed Description Text (80):
Any number of scripting languages could be devised to describe the connections. They could be specified in an easily human readable format, but this might be more difficult to produce automatically as shown later. In the preferred embodiment, the connections can be completely specified as data structures. That is, the scripting language chosen need only provide data, not programming instructions such as loops, if statements, etc. It could even be stored in a typical object-oriented or relational database. In fact, even a persistance model would be sufficient if parts of the model could be instantiated individually.

Detailed Description Text (81):
The invention as disclosed uses a hierarchical attributed data structure known as a table to store the meta-data that describes the classes and the connections between the classes. The data structure used is not as important as what is stored in it. Much can be learned about what data is required from the examples in Appendices F and G which contain the meta data for two relatively simple connections.

Detailed Description Text (86):
The edit function for an external object mapping is typically invoked programmatically, usually by the primary patron object (that stores the meta-data). In the preferred embodiment, edit is an inherited function from a base class. In C++, the call to edit would appear as:

Detailed Description Text (89):
1. The meta-data information for the object that the edit function is being invoked on is retrieved from the resource table using the name of the class. This takes advantage of the object's ability to be "self describing" at runtime.

Detailed Description Text (90):
2. A new instance of EosObjectViewMapper is created. This object is responsible for orchestrating the reading and interpreting of the meta-data and constructing each of the individual semantic links.

Detailed Description Text (92):
4. The table containing the connection descriptor (in this case "connectionName") is obtained from the meta-data database and is set on the new instance of EosObjectViewMapper. If the connection descriptor is not available in the meta-data for the class edit was invoked upon, then that class' superclass meta-data is retrieved and the descriptor is sought out there. This happens recursively until the base class is reached or the named connection is found. This is an important operation as it gives connections the same behavior as virtual functions. When the connections are composite view connections, this enables a unique technology known as "virtual views" where which view is used will change depending upon the type of the application class instance mapped to the view. If either the call to get the meta-data or to get the connection descriptor from the meta-data return invalid conditions the edit fails and NULL is returned.

Detailed Description Text (93):
5. The object view mapper object now traverses the table containing the connection descriptor. There is an entry for each semantic link. For each semantic link description, the individual mappers are created by name through the generic factory method and initialized using property information stored in the table for each semantic link. This includes the creation and initialization of the EosTypeElement members (fLeftSide and fRightSide) and their respective path objects. These like the mappers are created by type name according to the information stored in the descriptor information. The setProperties call allows the individual mappers associated with this connection to get the appropriate property information to perform the connection as it was set up in the builder. For all mappers there is at least:

Detailed Description Text (122):
(1) Select the primary class to which the connection will belong. This class will be the primary patron object involved in the connection and will also be the class in which the meta-data describing the connection will be stored.

Detailed Description Text (141):
The interface for creating composite view connections is much different than for the other kinds of connections disclosed. In this type of connection, the user's primary purpose is to build a user interface that has elements tied to language elements of his application objects. To create a composite view connection, he chooses to create a new connection, then chooses to create a composite view connection. After this, he types in a name for the connection, then chooses whether to create a window or a menu connection. If a window connection is chosen, a default view is created. Then the user selects either a field or function in the business object to which to connect the view object to. Depending upon the type of the field or function selected, a list of components (actually custom view connections) is built that the user can choose from. When a view is chosen, he chooses where to put it on the dialog being built, and may set visual properties on it (these properties are saved in the extensible meta-data format). The user has no direct access to the mapper properties at this point, but he may create a new custom view connection to do that.

Detailed Description Text (145):
The dialog described in step 2 above is a simple information gathering dialog. The information collected includes 1) The name to be given this collection of semantic links, that will later be used to look it up to instantiate the connection. 2) Whether this is an internal or external object mapping (or some other type of mapping). In the case of an internal object mapping, the only additional information required is the name of the class that the mapping is for. This is gathered from the context that the dialog was presented within. The primary class involved in the mapping must be selected before the mapping can be added to it. The primary class is the class that has the connection's meta-data stored within it. It would also be possible, of course, to simply enter this class type into a field of a dialog and free the interface from this contextual consideration.

Detailed Description Text (148):
Other connection types of connections are of course possible. In the preferred embodiment, composed user interfaces such as menus and dialogs, as well as custom connections to widgets, such as scroll bars, etc. are also selectable as connection types. These types of connections have more meta-data properties that describe the look and feel of the interface, but the basic connection of objects is exactly the same. Its just that in this special case, one side of the object connection is always a user interface class.

Detailed Description Text (149):
All of this information is stored in the meta-data by the code that brings up the dialog. The exact format of the meta-data is not as important as the information contained therein.

Detailed Description Text (152):
Accessing the meta-data database, the attributes, properties and functions that each object can be determined. These elements are added to the view as the user expands expandable nodes. Only the class attribute nodes are expandable unless a function or field has properties. As each node is expanded, the meta-data database is accessed, and the proper sub-tree added to the view. An icon is placed on each line of the tree to inform the user whether that line represents a property, a function, a primitive attribute or a class attribute.

Detailed Description Text (166):
Meta-Data

Detailed Description Text (167):
Meta-Data is data that describe data structures. In practice, meta-data lets an object be queried about its class data type, structure and functionality during program execution. For example, an object can be asked what its name is, how many functions it has, or what the name and type of the third field is. An object-oriented language that supports meta-data is said to have reflection, introspection or run-time type information (RTTI). C++ has a proposed RTTI extension, but it has not yet been implemented in most widely available compilers.

Detailed Description Text (168):
Meta-data itself is not a new concept. Whenever compilers scan code and create a symbol table as part of the compilation process, the symbol table is a meta-data database. This meta-data is commonly passed through the compilation process into the executable image of the program and is commonly used by debuggers during program development. However, in many languages, there is no standard way of accessing this information programmatically during program execution. In fact, as programs are prepared for final shipment, this debugging meta-data information is often removed from the program executable as it has no function in the final production program.

Detailed Description Text (169):
SmallTalk, LISP and other interpreted languages typically support meta-data access as an integral part of the language, although the amount and type of meta-data provided varies widely. Most compiled languages such as Pascal, Modula, C and C++ do not provide meta-data. There are of course notable exceptions to this general rule on both sides.

Detailed Description Text (170):
Meta-data is implemented class wide. That is, the data do not vary from object instance to object instance. Therefore, the meta-data can be stored efficiently in efficiently in one location. Also, meta-data is usually treated as read-only, and does not change during execution of the program. Interpreters provide an interesting exception when new code is generated, then executed by the same program without stopping. Dynamic binding and probes, on the other hand, are typically implemented on an instance basis rather than on a class basis.

Detailed Description Text (171):
The meta-data model as provided is extensible. Extensions to the meta-data are called properties. Properties can be applied to classes, fields, functions, mappers and other meta-data elements. The provision of properties allows for users of the system to add new functionality at the language level for various types. Mapper classes can respond to properties by various means. What this means is that a user can add properties to various data types and then respond to those properties within various mappers that he creates. This enables the programmer to enhance the mapper model with yet another degree of freedom. That is, in addition to adding properties to the mapper to establish behavior, properties can also be added to the elements being connected by the mapper.

Detailed Description Text (174):
Dynamic binding means that during execution of the program an object can be manipulated using the names of its members. For example, fields can be set and queried, and functions can be called by name. Any part of the program can invoke the member function func of class X during program execution on an instance of X without knowing anything but the name of the class and the function. Meta-data is useful in obtaining these names from the object itself

Detailed Description Text (176):
Dynamic binding is a fairly common feature in computer languages. Again, it is more common in interpreted languages and is often found in the same languages that support meta-data.

Detailed Description Text (177):
C++ has "dynamic binding" through the use of virtual functions. Virtual dynamic binding creates an array of function pointers for each class that has virtual functions. A pointer to this "Vtable" is stored in each object instance as it is constructed. This vtable pointer is used to invoke functions by index through the Vtable. Each subclasses can "override" virtual functions (causing a different function to be inserted in its local vtable) to implement different functionality. The determination of which function is called is therefore based upon the type of the object instance. U.S. Pat. Nos. 5,327,562 and 5,371,891 discuss virtual functions and their implementation in C++ type languages in explicit detail. The dynamic binding discussed herein is resolved at runtime using the names of the class members. Meta-data is required to do the form of dynamic binding discussed here, while it is not necessary to do the C++ type dynamic binding. C++ dynamic binding is strictly inheritance based, while the form discussed here is not. Only the specified functions of C++ classes are virtual, while all enhanced functions (including virtual functions) can be dynamically bound by the present invention.

Detailed Description Text (182):
Although meta-data and dynamic binding are implemented in SmallTalk, probes are not. The present invention could be applied to the enhancement of SmallTalk to support probes by one skilled in the art. Similarly, other languages that do not support probing could be enhanced using the present invention.

Detailed Description Text (186):
Some non-standard versions of Eiffel have a feature called "write barriers" that is also similar to probes. It allows for control over the assignment to variables. However, Eiffel lacks some of the meta-data and dynamic binding features discussed above.

Detailed Description Text (195):
Meta-Data for a Field to Field Connection

Detailed Description Text (196):
This meta-data contains the information required to map two EosInt fields of class ProjectFrame named "one" and "two" together as an internal object mapping. The mapper connecting the two fields is EosMapFieldToField.

Detailed Description Text (198):
Meta-Data for a Function Link

Detailed Description Text (199):
This is the meta-data specifying a EosMapMultiArgFunction link from

Other Reference Publication (1):
Huy Hguyen et al., OODDM: An Object-Oriented Database Design Model, Apr., 1990, see page 339, right-hand column, line 35; and page 341, left-hand column, line 2.

Previous Doc Next Doc Go to Doc#

Results for "((object oriented<in>metadata) <and> (relational<in>metadata))<and> (..."

Your search matched **490** of **1239820** documents.

A maximum of **100** results are displayed, **25** to a page, sorted by **Relevance** in **Descending** order.

» Search Options

View Session History

New Search

» Other Resources
(Available For Purchase)

Top Book Results

Emerging Trends in Database and Knowledge Based Machines
by Abdelguerfi, M.; Lavington, S.;
Paperback, Edition: 1

Meme Media and Meme Market Architectures
by Tanaka, Y.;
Hardcover, Edition: 1

View All 2 Result(s)

» Key

IEEE JNL	IEEE Journal or Magazine
IEE JNL	IEE Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IEE CNF	IEE Conference Proceeding
IEEE STD	IEEE Standard

Modify Search

((object oriented<in>metadata) <and> (relational<in>metadata))<and> (database

Check to search only within this results set

Display Format: Citation Citation & Abstract

Select Article Information

View: **1-25** | 26-50 | 51-75 | 76-100

1. **Query interoperation among object-oriented and relational databases**
Xiaolei Qian; Raschid, L.;
Data Engineering, 1995. Proceedings of the Eleventh International Conference on
6-10 March 1995 Page(s):271 - 278
Digital Object Identifier 10.1109/ICDE.1995.380384
AbstractPlus | Full Text: PDF(596 KB) IEEE CNF

2. **A formalization of object-oriented database and its functional query language**
Watanabe, T.; Tanemo, F.; Sugie, N.;
TENCON '94. IEEE Region 10's Ninth Annual International Conference. Theme: 'Frontiers of Computer Technology'. Proceedings of 1994
22-26 Aug. 1994 Page(s):511 - 515 vol.1
Digital Object Identifier 10.1109/TENCON.1994.369248
AbstractPlus | Full Text: PDF(408 KB) IEEE CNF

3. **Producing relational database schemata from an object oriented design**
Fitsilis, P.; Gerogiannis, V.; Kameas, A.; Pavlides, G.;
EUROMICRO 94. System Architecture and Integration. Proceedings of the 20th EUROMICRO Conference.
5-8 Sept. 1994 Page(s):251 - 257
Digital Object Identifier 10.1109/EURMIC.1994.390386
AbstractPlus | Full Text: PDF(420 KB) IEEE CNF

4. **Supporting data migration between relational and object-oriented databases using a federation approach**
Hohenstein, U.;
Database Engineering and Applications Symposium, 2000 International
18-20 Sept. 2000 Page(s):371 - 379
Digital Object Identifier 10.1109/IDEAS.2000.880614
AbstractPlus | Full Text: PDF(808 KB) IEEE CNF

5. **Federating object-oriented and relational databases: the IRO-DB experience**
Gardarin, G.; Finance, B.; Fankhauser, P.;
Cooperative Information Systems, 1997. COOPIS '97., Proceedings of the Second IFCIS International Conference on
24-27 June 1997 Page(s):2 - 13
Digital Object Identifier 10.1109/COOPIS.1997.613797
AbstractPlus | Full Text: PDF(1152 KB) IEEE CNF

6. **Capturing the objected-oriented database model in relational form**
Hsieh, S.-Y.; Chang, C.K.; Mongkolwat, P.; Pilch, W. W., Jr.; Shih, C.-C.;

Computer Software and Applications Conference, 1993. COMPSAC 93. Proceedings., Seventeenth Annual International
1-5 Nov. 1993 Page(s):202 - 208
Digital Object Identifier 10.1109/CMPSAC.1993.404191

AbstractPlus | Full Text: PDF(572 KB) IEEE CNF

7. **Measuring the contributions of (O)RDBMS to object-oriented software development**
Zhang, W.P.; Ritter, N.;
Database Engineering and Applications Symposium, 2000 International
18-20 Sept. 2000 Page(s):243 - 249
Digital Object Identifier 10.1109/IDEAS.2000.880583

AbstractPlus | Full Text: PDF(508 KB) IEEE CNF

8. **Implementation of object-oriented association relationships in relational databases**
Rahayu, W.; Chang, E.; Dillon, T.S.;
Database Engineering and Applications Symposium, 1998. Proceedings. IDEAS'98. International
8-10 July 1998 Page(s):254 - 263
Digital Object Identifier 10.1109/IDEAS.1998.694385

AbstractPlus | Full Text: PDF(64 KB) IEEE CNF

9. **An autonomous decentralized database system based on scripts and active objects**
Ishikawa, H.; Kubota, K.;
Autonomous Decentralized Systems, 1995. Proceedings. ISADS 95., Second International Symposium on
25-27 April 1995 Page(s):185 - 191
Digital Object Identifier 10.1109/ISADS.1995.398971

AbstractPlus | Full Text: PDF(728 KB) IEEE CNF

10. **An object-oriented prototype for a geophysical database**
Ramanathan, C.; Koduri, S.;
System Theory, 1995., Proceedings of the Twenty-Seventh Southeastern Symposium on
12-14 March 1995 Page(s):170 - 174
Digital Object Identifier 10.1109/SSST.1995.390596

AbstractPlus | Full Text: PDF(380 KB) IEEE CNF

11. **Queries in object-oriented databases**
Banerjee, J.; Kim, W.; Kim, K.-C.;
Data Engineering, 1988. Proceedings. Fourth International Conference on
1-5 Feb. 1988 Page(s):31 - 38
Digital Object Identifier 10.1109/ICDE.1988.105443

AbstractPlus | Full Text: PDF(648 KB) IEEE CNF

12. **Integration of relational databases in a multidatabase system based on schema enrichment**
Keim, D.A.; Kriegel, H.P.; Miethsam, A.;
Research Issues in Data Engineering, 1993: Interoperability in Multidatabase Systems, 1993. Proceedings RIDE-IMS '93., Third International Workshop on
19-20 April 1993 Page(s):96 - 104
Digital Object Identifier 10.1109/RIDE.1993.281939

AbstractPlus | Full Text: PDF(832 KB) IEEE CNF

13. **Association algebra: a mathematical foundation for object-oriented databases**
Su, S.Y.W.; Guo, M.; Lam, H.;
Knowledge and Data Engineering, IEEE Transactions on
Volume 5, Issue 5, Oct. 1993 Page(s):775 - 798
Digital Object Identifier 10.1109/69.243509

AbstractPlus | Full Text: PDF(2004 KB) IEEE JNL

14. **A prototype using a hybrid approach to recover dynamic semantics from relational schema**
Cheng, Y.W.;

Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings. 1999 IEEE International Conference on
Volume 5, 12-15 Oct. 1999 Page(s):750 - 755 vol.5
Digital Object Identifier 10.1109/ICSMC.1999.815645

AbstractPlus | Full Text: PDF(448 KB) IEEE CNF

15. **Poster on rule-based technology for schema transformation**
Yangjun Chen; Benn, W.;
Cooperative Information Systems, 1997. COOPIS '97., Proceedings of the Second IFCIS International Conference on
24-27 June 1997 Page(s):232
Digital Object Identifier 10.1109/COOPIS.1997.613825

AbstractPlus | Full Text: PDF(112 KB) IEEE CNF

16. **A type-safe object-oriented solution for the dynamic construction of queries**
Rosenthal, P.;
Data Engineering, 2004. Proceedings. 20th International Conference on
30 March-2 April 2004 Page(s):843
Digital Object Identifier 10.1109/ICDE.2004.1320073

AbstractPlus | Full Text: PDF(213 KB) IEEE CNF

17. **A temporal object oriented conceptual schema model**
Pasaya, B.; Chittayasothorn, S.;
Communications, Computers and signal Processing, 2001. PACRIM. 2001 IEEE Pacific Rim Conference on
Volume 2, 26-28 Aug. 2001 Page(s):724 - 727 vol.2
Digital Object Identifier 10.1109/PACRIM.2001.953734

AbstractPlus | Full Text: PDF(256 KB) IEEE CNF

18. **ORR: object-relational rapprochement**
Crigler, J.B.; Orooji, A.;
Computer Software and Applications Conference, 1999. COMPSAC '99. Proceedings. The Twenty-Third Annual International
27-29 Oct. 1999 Page(s):42 - 48
Digital Object Identifier 10.1109/CMPSAC.1999.812674

AbstractPlus | Full Text: PDF(588 KB) IEEE CNF

19. **Object life-cycles in active relational databases**
Porto, F.; Vianna e Silva, M.J.; Carvalho, S.;
Technology of Object-Oriented Languages, 1998. TOOLS 26. Proceedings
3-7 Aug. 1998 Page(s):168 - 179
Digital Object Identifier 10.1109/TOOLS.1998.711011

AbstractPlus | Full Text: PDF(60 KB) IEEE CNF

20. **A design environment for migrating relational to object oriented database systems**
Jahnke, J.; Schafer, W.; Zundorf, A.;
Software Maintenance 1996, Proceedings., International Conference on
4-8 Nov. 1996 Page(s):163 - 170
Digital Object Identifier 10.1109/ICSM.1996.565001

AbstractPlus | Full Text: PDF(624 KB) IEEE CNF

21. **Using object-oriented principles to optimize update propagation to materialized views**
Kuno, H.A.; Rundensteiner, E.A.;
Data Engineering, 1996. Proceedings of the Twelfth International Conference on
26 Feb.-1 March 1996 Page(s):310 - 317
Digital Object Identifier 10.1109/ICDE.1996.492178

AbstractPlus | Full Text: PDF(828 KB) IEEE CNF

22. **Transformation of relational schemas to object-oriented schemas**
Weiyi Meng; Kamada, A.; Yu-Hsi Chang;
Computer Software and Applications Conference, 1995. COMPSAC 95. Proceedings., Nineteenth Annual International

9-11 Aug. 1995 Page(s):356 - 361
Digital Object Identifier 10.1109/CMPSAC.1995.524801

AbstractPlus | Full Text: PDF(628 KB) IEEE CNF

23. **Issues in the design and implementation of views in object-oriented databases**
Venugopal, V.;
Computers and Communications, 1990. Conference Proceedings., Ninth Annual International Phoenix Conference on
21-23 March 1990 Page(s):903
Digital Object Identifier 10.1109/PCCC.1990.101741

AbstractPlus | Full Text: PDF(52 KB) IEEE CNF

24. **Integration of database systems and Smalltalk**
Czejdo, B.; Taylor, M.C.;
Applied Computing, 1991., [Proceedings of the 1991] Symposium on
3-5 April 1991 Page(s):393 - 402
Digital Object Identifier 10.1109/SOAC.1991.143909

AbstractPlus | Full Text: PDF(636 KB) IEEE CNF

25. **An object-oriented query language interface to relational databases in a multidatabase database environment**
Urban, S.D.; Abdellatif, T.B.;
Distributed Computing Systems, 1994., Proceedings of the 14th International Conference on
21-24 June 1994 Page(s):387 - 394
Digital Object Identifier 10.1109/ICDCS.1994.302441

AbstractPlus | Full Text: PDF(680 KB) IEEE CNF

View: **1-25** | 26-50 | 51-75 | 76-100

Federating object-oriented and relational databases: the IRO-DB experience

Gardarin, G. Finance, B. Fankhauser, P.
Lab. PRiSM URA CNRS 1525, Univ. de Versailles-St. Quentin, Versailles, France;

This paper appears in: **Cooperative Information Systems, 1997. COOPIS '97., Proceedings of the Secor IFCIS International Conference on**
Publication Date: 24-27 June 1997
On page(s): 2 - 13
Number of Pages: xiii+233
Meeting Date: 06/24/1997 - 06/27/1997
Location: Kiawah Island, SC
INSPEC Accession Number:5651835
Digital Object Identifier: 10.1109/COOPIS.1997.613797
Posted online: 2002-08-06 21:21:36.0

Abstract
From the beginning of 1994 to the end of 1996, the IRO-DB (Interoperable **Relational** and **Object-Oriented Databases**) ESPRIT project has developed tools for accessing **relational** and **object-oriented databases** i an integrated way, and for designing and maintaining integrated applications on large federations of heterogeneous databases. IRO-DB is based on the ODMG pivotal object model and gives an OQL/OML-C++ interface to users on a federation of **relational** and **object-oriented** databases. This paper summarizes the main problems and choices done during the system design, describes the IRO-DB architecture and components, presents the project's achievements and gives a synthesis of the lessons learned during the project's development. It also introduces future plans for the system

Index Terms
Inspec
Controlled Indexing
distributed databases integrated software object-oriented databases open systems relational databases research initiatives user interfaces

Non-controlled Indexing
ESPRIT project IRO-DB project ODMG object model OQL/OML-C++ interface database access tools database architecture database components federated heterogeneous databases integrated applications interoperable databases object-oriented databases relational databases system design

Author Keywords
Not Available

References

No references available on IEEE Xplore.

Citing Documents

1 A three-tier view-based methodology for M-services adaptation, Chiu, D.K.W.; Cheung, S.C.; Kafeza, E.; Ho-Fung Leung
Systems, Man and Cybernetics, Part A, IEEE Transactions on
On page(s): 725- 741, Volume: 33, Issue: 6, Nov. 2003
Abstract | Full Text: PDF (2056)